Thinking Without Speaking: Efficient Tree of Thoughts Reasoning via Latent Steering

Anonymous ACL submission

Abstract

Tree of Thoughts (ToT) reasoning is fundamentally bottlenecked by its generate-then-score paradigm, which decodes every candidate branch into text and evaluates it through a decoupled scoring stage. Prior efficiency efforts primarily shrink the search breadth, leaving this prohibitive per-step textual evaluation intact. To resolve this, we propose **IToT** (Implicit Tree of Thoughts), a training-free framework that recasts tree-structured reasoning as a continuous search within a model's latent space. In this framework, branch quality is assessed intrinsically from hidden states via alignment with a contrastive prototype distilled from successful and failed traces. This elegantly collapses explicit branch enumeration and textual scoring into a single latent generate-evaluate-decide loop. Across six reasoning benchmarks and five LLM backbones, IToT consistently improves mean accuracy over both the strongest tree-search and single-path baselines by roughly ten absolute points. Crucially, it achieves this while cutting token consumption by nearly an order of magnitude relative to explicit tree-search methods, operating even below the token budget of vanilla Chain of Thought. We will release our code to facilitate further research.

1 Introduction

While the Tree of Thoughts (ToT) framework enables large language models (LLMs) to tackle complex tasks via tree-structured exploration (Yao et al., 2023; Plaat et al., 2025), its deployment is hindered by prohibitive inference costs. A fundamental bottleneck is the explicit decoupling of candidate generation and evaluation (Yao et al., 2023; Madaan et al., 2023; Lightman et al., 2023; Besta et al., 2024). Current variants typically rely on a generate-then-score pipeline, invoking independent verification at each expansion step. Coupled with nontrivial branching factors and search depths, this per-node overhead compounds rapidly, severely degrading throughput and rendering ToT impractical for large-scale or latency-sensitive applications (Wang et al., 2025a,b).

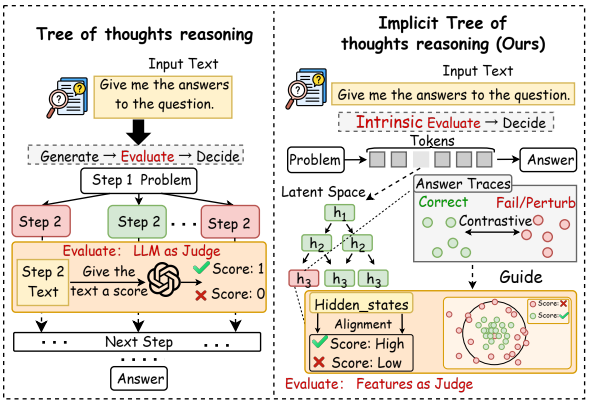

Figure 1: **Explicit evaluation versus implicit latent-space guidance.** Unlike Tree-of-Thoughts, which expands and scores candidate steps in text, our framework selects branches directly in latent space using hidden-state differences between effective and ineffective traces, absorbing evaluation into generation.

Existing efficiency improvements for ToT primarily focus on restricting the explored search space via branch pruning (Kim et al., 2025; Gao et al., 2026), constrained expansion (Ding et al., 2025), or state summarization (Yang et al., 2024b). However, these strategies fail to resolve the underlying architectural flaw. By continuing to rely on separate evaluation signals for generated candidates (Hao et al., 2023a; Besta et al., 2024), such methods reduce the total number of expanded nodes but leave the critical per-step latency of the generate-then-score mechanism entirely intact.

To address this bottleneck at the architectural level, we integrate evaluation into the generation process itself, thereby eliminating the need for explicit text-level scoring as a separate stage (Hao et al., 2024; Zhang et al., 2025b). This design reduces latency because candidate branches no longer need to be generated first and then assessed through

auxiliary textual judgments or additional evaluation passes. Instead, branch quality is reflected directly in the model's internal states during generation. Inspired by prior findings that effective and ineffective reasoning traces exhibit systematic differences in the internal states of large language models (Wang et al., 2024; Vilas et al., 2025), we directly exploit these differences as guidance signals for reasoning. As illustrated in Figure 1, our method replaces the explicit generate-then-score loop with an intrinsic generate-evaluate-decide process operating in latent space: hidden representations associated with more promising reasoning paths are encouraged, while those associated with less reliable continuations are suppressed. As a result, structured multi-trajectory reasoning can proceed without explicit textual judging, substantially reducing the per-step computational overhead characteristic of prior generate-then-score frameworks (Zelikman et al., 2024; Sun et al., 2026).

Building upon this insight, we introduce Implicit Tree of Thoughts (IToT) to eliminate the inference bottlenecks of traditional multi-step reasoning. IToT solves the latency issue by abandoning discrete textual branching in favor of continuous latent optimization. By conceptualizing the generation process as a latent tree search, IToT replaces costly token-level branching with CMA-ES (Hansen, 2016) perturbations, where decoding windows natively dictate the search depth.

To overcome the prohibitive overhead of text-level scoring, we propose IToT, a framework that integrates evaluation directly into continuous latent search. Rather than halting generation to score explicit text, IToT maintains a population of latent steering directions anchored to a contrastive prototype. At each decoding window, candidate directions are intrinsically evaluated via their alignment with this prototype, refined through population-based search, and softly injected back into the hidden states. Stabilized by a momentum update, this unified procedure preserves the powerful search capabilities of ToT while structurally eradicating the delays of explicit verification.

Empirical results across six benchmarks and five backbones show that IToT establishes a new state-of-the-art in reasoning efficiency. It surpasses the mean accuracy of existing explicit-search methods (e.g., ToT, GoT, DPTS) by a staggering 28.4 points. Most notably, IToT achieves these robust multi-trajectory reasoning capabilities while reducing token usage by roughly $10\times$ relative to explicit search, requiring even fewer tokens than a standard single-path CoT.

In summary, our core contributions are threefold: **First**, we introduce a contrastively calibrated latent quality proxy that captures the geometric signature of successful reasoning. **Second**, we formulate tree-structured reasoning entirely as a continuous latent optimization problem, mapping discrete tree depth and branches to generation windows and directional perturbations. **Finally**, we instantiate this formulation via a highly efficient inference-time controller, leveraging latent beam selection and momentum accumulation, to deliver the reasoning benefits of explicit search without its prohibitive token generation costs.

2 Motivation and Problem Formulation

The efficiency bottleneck in ToT reasoning stems not just from search breadth, but significantly from the cost of branch evaluation. In standard generate-then-score pipelines, every candidate reasoning step must be decoded into text and evaluated by a separate procedure. This incurs a substantial per-branch overhead that persists even when pruning restricts the search breadth. Overcoming this bottleneck requires fundamentally rethinking whether branch quality must be assessed via explicit text generation. *If the model's internal activations already reflect useful quality signals during generation, we can perform branch selection directly in the latent space, bypassing the overhead of auxiliary text decoding.*

This hypothesis rests on two key observations. First, upper-layer hidden states in large language models capture abstract, decision-relevant semantic information, in contrast to lower layers which focus on local lexical and syntactic processing (Liu et al., 2019; Chen et al., 2023; Skean et al., 2025). Second, successful and unsuccessful reasoning traces exhibit systematic divergences in their hidden-state trajectories (Wang et al., 2024; Zhang et al., 2025a; Vilas et al., 2025). Crucially, these divergences are often detectable at intermediate decoding steps (Vilas et al., 2025). Consequently, we are motivated to replace discrete, text-level branch evaluation with lightweight, continuous latent-space guidance.

2.1 Preliminaries

Let $f_\theta = f_\theta^{(L)} \circ \cdots \circ f_\theta^{(1)}$ be a pretrained decoder-only LLM with L Transformer blocks, and let $\mathbf{h}_t^{(l)}$ denote its hidden state at decoding step t and layer

l for an input x and an autoregressively generated response $y = (y_1, \ldots, y_T)$. Since upper layers encode abstract task-level semantics, we restrict analysis and intervention to $\mathcal{L}_{\text{high}} = \{l_{\text{start}}, \ldots, L\}$, and use the final-token state of a completed trace,

$$\mathbf{z}^{(l)}(x, y) \;=\; \mathbf{h}_T^{(l)}(x, y), \qquad l \in \mathcal{L}_{\text{high}},$$

as its response-level latent representation, while end-of-window states $\mathbf{h}_{t_w}^{(l)}$ are used online for branch control.

Given a calibration set $\mathcal{D}_{\text{cal}} = \{(x_i, y_i^+, y_i^-)\}_{i=1}^N$ of paired successful/failed traces, we average the failure-to-success direction to obtain a per-layer contrastive prototype

$$\mathbf{p}^{(l)} = \text{Norm}\Big(\sum_{i=1}^{N} (\mathbf{z}^{(l)}(x_i, y_i^+) - \mathbf{z}^{(l)}(x_i, y_i^-))\Big),$$

where $\text{Norm}(\cdot)$ denotes ℓ_2 normalization (Wang et al., 2024; Vilas et al., 2025). We use $\mathbf{p}^{(l)}$ not as a verifier but as an efficient quality proxy: its alignment with a candidate's hidden state serves as the branch-evaluation score, removing the need for explicit text decoding. The full derivation, the choice of l_{start}, and the calibration protocol are deferred to Appendix A.

2.2 Problem Formulation

Having established the latent prototype as a continuous quality proxy, we now formulate the reasoning process itself as an optimization problem over these hidden states. Normally, once its prefix is fixed, an autoregressive LLM follows a single, static latent trajectory. Our goal is to transform this default behavior into a structured, tree-like search over latent steering directions, operating strictly with frozen model parameters and avoiding explicit text-level evaluation.To achieve this, we partition the decoding process into fixed-length windows of W tokens. At a given window w, a controller proposes K candidate steering directions:

$$\mathcal{V}_w = \{\mathbf{v}_{w,1}, \ldots, \mathbf{v}_{w,K}\}, \qquad \mathbf{v}_{w,k} = \{\mathbf{v}_{w,k}^{(l)}\}_{l \in \mathcal{L}_{\text{high}}}.$$

Here, each $\mathbf{v}_{w,k}$ instantiates a candidate latent branch within the controlled layers. We then score these branches based on their cosine alignment with our pre-calculated prototype directions:

$$s(\mathbf{v}_{w,k}) = \sum_{l \in \mathcal{L}_{\text{high}}} \omega_l \, \cos(\mathbf{v}_{w,k}^{(l)}, \mathbf{p}^{(l)}),$$

where $\omega_l \geq 0$ represents optional layer weights. The direction yielding the highest score is selected and continuously mixed into the model's hidden states to guide the subsequent decoding window.Crucially, this formulation maps traditional ToT directly into the latent space: decoding windows dictate tree depth, proposed steering directions act as alternative search branches, and latent prototype alignment serves as the branch evaluation mechanism. The defining advantage over standard ToT is that branch selection occurs intrinsically within the hidden representations. Given an input x, a frozen model f_θ, and the prototype set $\mathbf{P} = \{\mathbf{p}^{(l)}\}_{l \in \mathcal{L}_{\text{high}}}$, our controller π yields the output:

$$\hat{y} = \text{Decode}_\pi(x; f_\theta, \mathbf{P}).$$

Our objective is to find the optimal controller policy $\pi^\star$ that maximizes the expected reward:

$$\pi^\star = \arg\max_{\pi \in \Pi} E_{(x, y^\star) \sim \mathcal{D}} \left[R(\hat{y}, y^\star)\right],$$

constrained by frozen model parameters and interventions limited to upper-layer hidden states. Ultimately, by executing branch evaluation entirely within the latent space, this framework guarantees that the only tokens explicitly decoded are those strictly belonging to the final generated reasoning path.

3 Method

3.1 Overview

As illustrated in Figure 2, we propose **IToT**, a training-free framework that enhances LLM reasoning by steering internal hidden states, without emitting any explicit intermediate reasoning tokens. IToT operates in two phases. Offline, *Prototype Calibration* collects hidden states from a small set of model-generated traces labeled as logically correct or confused, and contrasts them to derive a prototypical direction that captures the geometric gap between coherent and incoherent reasoning. Online, *Latent Steering* maintains multiple candidate reasoning branches entirely in the hidden-state space: at each step, branches are scored by their alignment with the prototypical direction, and the top branch steers the next token. Every W tokens, CMA-ES refines the branch population from this intrinsic signal, letting the search adapt as generation proceeds. By performing tree-structured search implicitly in latent space, IToT attains multi-path exploration without verbose intermediate output,

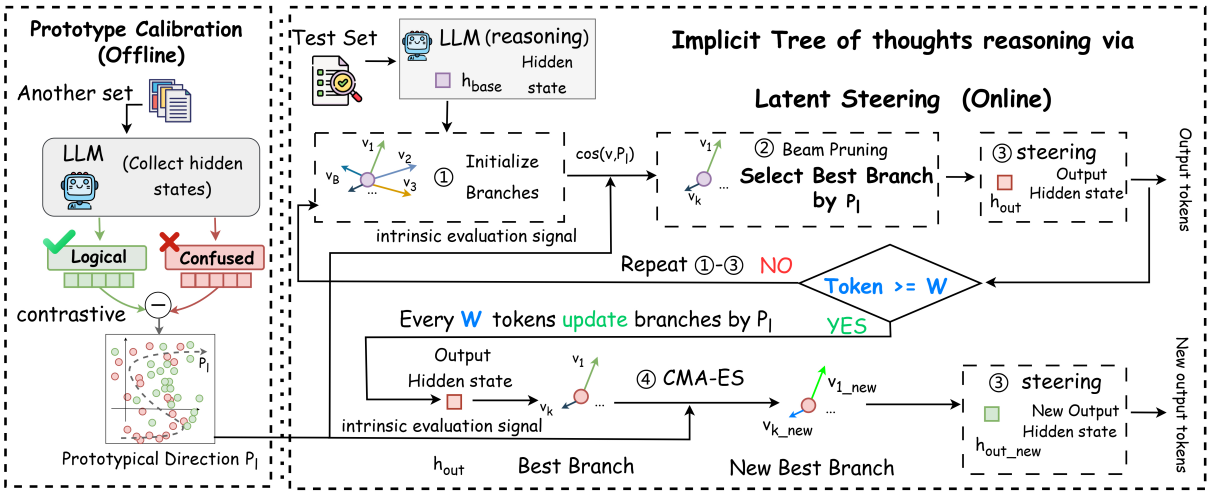

Figure 2: Overview of IToT. Offline calibration (left) derives a prototypical direction $\mathbf{P}_l$ via contrastive learning on hidden states. During online inference (right), beam pruning selects the best reasoning branch by $\cos(v, \mathbf{P}_l)$, and CMA-ES updates branches every W tokens to steer the model toward more logical outputs.

remaining both efficient and transparent to downstream decoding.

3.2 Offline Prototype Calibration

The objective of offline calibration is to distill the abstract concept of "correct reasoning" into a geometric vector within the model's representation space. Rather than relying on external reward models or explicitly prompting the model to verify text, IToT isolates a latent *steering direction* that consistently points from faulty logic toward accurate problem-solving. This direction acts as an intrinsic geometric compass, enabling the model to self-evaluate the quality of its intermediate reasoning states instantly, without the overhead of generating auxiliary evaluation tokens.

Formally, to construct this quality proxy, IToT computes a contrastive prototype direction $\mathbf{p}^{(l)}$ for each controlled upper layer $l \in \mathcal{L}_{\text{high}}$. Given a calibration set of inputs x_i, we construct strictly paired successful and failed traces, denoted as (y_i^+, y_i^-). The successful trace y_i^+ is derived directly from the reference solution. To rigorously isolate the latent signature of reasoning quality from confounding topical or stylistic variations, we synthesize the failed trace y_i^- by prompting the frozen model to generate a verbose, mistake-prone derivation under high-temperature stochastic decoding. By contrasting strictly paired responses to the same prompt, the semantic features of the specific question are effectively canceled out, leaving a pure "reasoning delta". The resulting layer-wise prototypes $\{\mathbf{p}^{(l)}\}_{l \in \mathcal{L}_{\text{high}}}$ are aggregated offline and ℓ_2-normalized. During inference, these prototypes remain strictly frozen, functioning as a computationally lightweight scoring signal that evaluates candidate latent branches via cosine alignment in the hidden-state space.

3.3 Periodic Latent Branching and Selection

To execute the latent search process, IToT implements a factorized, layer-wise controller. Rather than maintaining a monolithic global state, each controlled layer $l \in \mathcal{L}_{\text{high}}$ independently manages a branch pool:

$$\left\{(\mathbf{m}_b^{(l)},\, v_b^{(l)})\right\}_{b=1}^{B},$$

where $\mathbf{m}_b^{(l)} \in R^d$ represents a unit-norm latent steering direction (the branch) and $v_b^{(l)} \in R$ tracks its accumulated quality score (the branch value). To balance exploration with reasoning relevance, this pool is initialized by applying normalized Gaussian perturbations to the calibrated prototype. This ensures that all candidate branches originate within a promising region of the latent space while maintaining the diversity necessary for search. The latent tree search unfolds dynamically over fixed-length generation windows of W tokens. At the end of each window, the framework executes three core operations to evolve the latent branches: Branch Expansion (CMA-ES): To explore alternative reasoning paths, every surviving branch undergoes local, derivative-free refinement. We employ CMA-ES (Hansen, 2016) to evolve the current branch direction with additional noise, searching for a new

unit vector $\mathbf{d}_b^{(l)}$ that maximizes cosine alignment with the layer-wise prototype $\mathbf{p}^{(l)}$. Momentum Update and Scoring: To stabilize the search across long reasoning horizons and prevent erratic steering, the newly discovered direction $\mathbf{d}_b^{(l)}$ is integrated into the branch's historical trajectory using a momentum update:

$$\mathbf{m}_b^{(l)} \leftarrow \mathrm{Norm}\Big(\beta \mathbf{m}_b^{(l)} + (1-\beta)\mathbf{d}_b^{(l)}\Big),$$

where $\beta \in [0,1]$ is the momentum coefficient. Simultaneously, the branch value is updated by accumulating the alignment reward derived from the newly refined direction:

$$v_b^{(l)} \leftarrow v_b^{(l)} + \cos(\mathbf{d}_b^{(l)}, \mathbf{p}^{(l)}).$$

Latent Pruning: Once all branches are expanded and scored, IToT mimics the selection phase of traditional ToT. Only the top-K branches, ranked strictly by their accumulated value $v_b^{(l)}$, are retained for the next decoding window.Through this continuous cycle, IToT elegantly maps discrete tree concepts to latent operations: decoding windows dictate tree depth, directional refinement instantiates branch expansion, and alignment-based pruning completely replaces textual scoring.

3.4 Continuous Latent Steering

Identifying the optimal latent branch is only half of the solution; the framework must also execute this branch by actively guiding the model's token generation. The primary challenge here is intervention stability: aggressively overwriting hidden states can easily derail the LLM's foundational linguistic and syntactic capabilities. To mitigate this, IToT employs a continuous "soft steering" mechanism. This design delicately injects the identified reasoning bias into the model's forward pass, striking a rigorous balance between shifting the semantic trajectory and strictly preserving the local, token-level decoding dynamics.

Between the windowed branch-update points, IToT continuously modulates the generation process using the highest-scoring branch at each controlled layer. For a given layer l, the optimal branch index is selected based on the accumulated branch value:

$$b^{*(l)} = \arg\max_b v_b^{(l)}.$$

Let $\mathbf{h}_t^{(l)}$ denote the model's hidden state at decoding step t and layer l. To ensure stable intervention, we specifically target the post-attention, layer-normalized representation. We first construct a directionally biased state. Crucially, to prevent the intervention from pushing the activation out of its natural manifold, we scale the chosen unit steering direction $\mathbf{m}_{b^{*(l)}}^{(l)}$ by the ℓ_2 norm of the current hidden state:

$$\tilde{\mathbf{h}}_t^{(l)} = \mathbf{h}_t^{(l)} + \lambda^{(l)} \|\mathbf{h}_t^{(l)}\|_2 \mathbf{m}_{b^{*(l)}}^{(l)},$$

where $\lambda^{(l)}$ dictates the layer-specific steering strength. Rather than replacing the original hidden state entirely, we apply a soft convex mixture:

$$\mathbf{h}_t'^{(l)} = (1-\alpha)\mathbf{h}_t^{(l)} + \alpha \tilde{\mathbf{h}}_t^{(l)},$$

with the mixture coefficient $\alpha \in [0,1]$.The modified representation $\mathbf{h}_t'^{(l)}$ is then passed to the subsequent layers. Through this precise mathematical formulation, IToT seamlessly translates the winning latent branch into a stable, continuous reasoning bias, completely bypassing explicit textual manipulation. The full execution pipeline of IToT is summarized in Algorithm 1 in the Appendix B.

4 Experiment

4.1 Settings

Datasets. We select six datasets across four critical dimensions of LLM capabilities (Zheng et al., 2023; Huang et al., 2023; Wang et al., 2024): (1) Mathematics: GSM8K (Cobbe et al., 2021) and MATH (Hendrycks et al., 2021); (2) Reasoning: CommonsenseQA (Talmor et al., 2019) and PROSQA (Hao et al., 2024); (3) Knowledge: MMLU (Hendrycks et al., 2020); and (4) Understanding: Belebele (Bandarkar et al., 2024). Dataset details are provided in Appendix C.1.

Language Models. Our evaluation includes Gemma-3-1B-IT (Team et al., 2025), DeepSeek-R1-Distill-Qwen-1.5B (Guo et al., 2025), Qwen2.5-1.5B-Instruct (Qwen et al., 2025), Qwen2.5-3B-Instruct (Qwen et al., 2025), and Qwen2.5-Math-1.5B-Instruct (Yang et al., 2024a). Additional details are provided in Appendix C.2.

Baselines. We compare our approach against six explicit reasoning baselines, which require models to generate intermediate reasoning steps : Chain-of-Thought (CoT) (Wei et al., 2022), Self-Consistency with CoT (CoT-SC) (Wang et al., 2022), Tree-of-Thoughts (ToT) (Yao et al., 2023), Graph-of-Thoughts (GoT) (Besta et al., 2024), DPTS (Ding et al., 2025), and SSDP (Kim et al., 2025). Additional comparisons are detailed in C.3 and with

implicit reasoning methods (e.g., Coconut (Hao et al., 2024) and CODI (Shen et al., 2025)) are detailed in Appendix C.5.

Implementation and Evaluation. All experiments are conducted on 48G GPUs, and we report the average results of 5 runs by default. We assess both effectiveness and efficiency using three complementary metrics : (1) *accuracy*, which measures the proportion of correctly predicted outputs; (2) *runtime per sample*, capturing the average inference time under identical hardware settings; and (3) *total token consumption per sample*, which quantifies the overall inference cost by counting all generated tokens. Additional implementation details are provided in Appendix C.4.

4.2 Main Results

Table 1 reports accuracy, runtime, and total token consumption of IToTagainst six representative baselines across five backbones and six datasets. We summarize the key findings below.

IToT achieves the best average accuracy on 5 out of 6 datasets. Compared with the strongest baseline on each dataset, IToT improves the average accuracy from 0.636 (ToT) to **0.701** on MATH500 (+6.5 pts), from 0.734 (CoT) to **0.757** on CommonsenseQA, from 0.763 (CoT) to **0.775** on Bele-bele, from 0.668 (CoT) to **0.689** on MMLU, and most strikingly from 0.469 (ToT) to **0.812** on ProsQA (+34.3 pts). On GSM8K our method (0.853) is on par with CoT (0.871), suggesting that for short-horizon arithmetic problems a single chain is already sufficient and structured search yields diminishing returns.

The gain is largest on tasks that require structured, multi-hop reasoning. ProsQA is the clearest stress test: tree- and graph-based baselines either collapse (DPTS 0.097, SSDP 0.057, GoT 0.338) or suffer from extreme token blow-up (ToT consumes ∼41K tokens per sample). IToT reaches 0.812 average accuracy using only 3,413 tokens per sample, 12× fewer than ToT and 20× fewer than DPTS. This indicates that the proposed mechanism prunes unproductive branches effectively while preserving the reasoning paths that matter, which is precisely where existing search-based methods over-explore.

Token efficiency is competitive with single-chain methods and an order of magnitude better than search-based baselines. Across the six datasets, IToT consumes on average 1,544–3,413 tokens per sample, comparable to CoT (1,517–4,057) and 4–20× smaller than ToT, DPTS, and SSDP. For example, on CommonsenseQA IToT uses 1,874 tokens vs. 14,159 for DPTS (7.5× reduction) while improving accuracy by 33.7 points. This confirms that the accuracy gains do *not* come from larger compute budgets but from a more selective expansion of the reasoning space.

Runtime is moderate and predictable. IToT runs faster than ToT and GoT on almost every dataset (e.g., 90.0s vs. 178.8s and 416.1s on ProsQA) and is within 1.0–1.5× of CoT. Although DPTS and SSDP can be faster in wall-clock time, this is achieved at a substantial cost in accuracy (e.g., 0.097 and 0.057 on ProsQA), revealing that their early-termination strategies sacrifice correctness on hard, long-horizon problems.

Robustness across backbones. Search-based baselines degrade sharply on weaker or mismatched backbones. On Gemma-3-1B, DPTS and SSDP fall to 0.135/0.063 on GSM8K and 0.081/0.024 on MMLU, indicating brittle pruning under unreliable value estimates. In contrast, IToT retains 0.813 on GSM8K and 0.464 on MMLU, demonstrating strong backbone robustness and graceful scaling to smaller models.

The results show that IToT combines the accuracy of structured search with the efficiency of single-chain prompting. Gains are largest on multi-step reasoning tasks (MATH500, ProsQA) and consistent across five diverse backbones, including a 1B and a math-specialized model.

4.3 Ablation Experiments

Table 2 details the ablation study of IToT on the GSM8K benchmark. The full framework achieves the highest accuracy (0.871), validating the synergy of its integrated components. The results highlight two primary structural drivers of success: latent tree expansion and the steering mechanism. Removing the tree causes the most substantial accuracy drop (0.871 → 0.859), indicating that hierarchical multi-trajectory exploration is essential for maintaining a robust candidate set. Similarly, disabling the steering mechanism degrades performance, confirming that active latent-space guidance is indispensable for shaping reliable reasoning trajectories. Beyond this structural backbone, the continuous optimization modules deliver consistent, complementary gains. Ablating CMA,

Table 1: Main results on accuracy and efficiency across six datasets. Each entry reports accuracy ↑, runtime per sample ↓, and total token consumption per sample ↓. Accuracy and runtime are reported to three decimal places, while token consumption is reported to two decimal places. underline represents the SOTA performance among all baselines, **bold** represents the SOTA performance among all methods.

Method / Model	Qwen2.5-1.5B	Qwen2.5-3B	DeepSeek-R1-Distill-Qwen-1.5B	Qwen2.5-Math-1.5B	Gemma-3-1B	Avg.
GSM8K (Accuracy ↑ / Runtime ↓ / Total Token ↓)						
1. CoT	0.838 / 21.078 / **1,502.80**	**0.945** / 38.524 / **1,408.58**	0.870 / 107.774 / 4,685.80	0.917 / 41.598 / 1,848.16	0.783 / 52.887 / **2,341.38**	**0.871** / 52.372 / **2,357.34**
2. CoT-SC	0.672 / 21.078 / **1,502.80**	0.882 / 38.524 / **1,408.58**	0.700 / 107.774 / 4,685.80	0.823 / 41.598 / 1,848.16	0.667 / 52.887 / **2,341.38**	0.749 / 52.372 / **2,357.34**
3. ToT	0.729 / 50.859 / 25,775.44	0.777 / 13.989 / 2,341.48	**0.892** / 108.455 / 59,641.34	**0.922** / 45.914 / 6,428.81	0.405 / **13.918** / 20,444.41	0.745 / 46.627 / 22,926.30
4. GoT	0.823 / 60.370 / 2,403.37	0.889 / 88.105 / 2,610.90	0.662 / 93.680 / **3,673.28**	0.872 / 61.163 / 2,571.65	0.627 / 656.690 / 4,689.27	0.775 / 192.002 / 3,189.69
5. DPTS	0.738 / 10.586 / 7,035.33	0.847 / **12.969** / 8,396.50	0.704 / 29.668 / 32,710.00	0.832 / **9.992** / 7,034.33	0.135 / 17.050 / 6,682.27	0.651 / 16.053 / 12,371.69
6. SSDP	0.624 / **9.924** / 6,029.43	0.799 / 13.567 / 6,106.24	0.653 / **23.791** / 15,832.76	0.858 / 10.157 / 5,716.12	0.063 / 16.950 / 6,208.03	0.599 / **14.878** / 7,978.52
IToT (Ours) ∗	**0.871** / 25.690 / 1,505.67	0.927 / 45.453 / 1,499.78	0.755 / 62.079 / 4,641.32	0.898 / 33.922 / **1,768.35**	**0.813** / 63.168 / 3,134.33	0.853 / 46.062 / 2,509.89
MATH500 (Accuracy ↑ / Runtime ↓ / Total Token ↓)						
1. CoT	0.634 / 64.258 / 2,764.72	0.740 / 94.334 / 3,101.58	0.422 / 120.512 / **4,618.40**	0.782 / 69.000 / 2,874.26	0.524 / 56.370 / **3,114.32**	0.620 / 80.895 / **3,294.66**
2. CoT-SC	0.492 / 64.258 / 2,764.72	0.642 / 94.334 / 3,101.58	0.251 / 120.512 / **4,618.40**	0.706 / 69.000 / 2,874.26	0.414 / 56.370 / **3,114.32**	0.501 / 80.895 / **3,294.66**
3. ToT	0.554 / 99.218 / 31,917.77	0.592 / 37.551 / 5,422.34	**0.808** / 550.928 / 72,533.27	0.742 / 129.105 / 15,276.52	0.482 / 48.712 / 27,084.19	0.636 / 173.103 / 30,446.82
4. GoT	0.606 / 130.506 / 3,727.25	0.616 / 135.275 / 3,876.19	0.698 / 376.689 / 13,870.07	0.664 / 98.314 / 3,439.56	0.482 / 627.510 / 7,119.07	0.613 / 273.659 / 6,406.43
5. DPTS	0.474 / **23.546** / 12,005.25	0.514 / 27.035 / 13,900.75	0.417 / 66.337 / 90,608.50	0.664 / **19.375** / 11,328.00	0.336 / **39.114** / 12,311.00	0.481 / **35.081** / 28,030.70
6. SSDP	0.491 / 25.890 / 17,386.65	0.533 / **25.857** / 13,853.69	0.559 / **65.157** / 62,709.25	0.678 / 19.471 / 8,151.95	0.359 / 53.520 / 9,551.65	0.524 / 37.979 / 22,330.64
IToT (Ours) ∗	**0.726** / 50.072 / **2,714.33**	**0.794** / 95.237 / **2,838.08**	0.538 / 87.269 / 4,925.48	**0.814** / 53.029 / **2,792.85**	**0.631** / 63.391 / 3,521.23	**0.701** / 69.800 / 3,358.39
COMMONSENSEQA (Accuracy ↑ / Runtime ↓ / Total Token ↓)						
1. CoT	**0.877** / 24.166 / 3,817.10	0.865 / 50.922 / **1,324.14**	0.726 / 70.672 / **3,100.54**	0.635 / 81.410 / 3,355.34	0.569 / 8.842 / **671.44**	0.734 / 47.202 / 2,453.71
2. CoT-SC	0.735 / 24.166 / 3,817.10	0.718 / 50.922 / **1,324.14**	0.393 / 70.672 / **3,100.54**	0.332 / 81.410 / 3,355.34	0.452 / 8.842 / **671.44**	0.526 / 47.202 / 2,453.71
3. ToT	0.832 / 28.775 / 9,792.29	0.779 / **2.386** / 1,572.44	0.655 / 132.896 / 7,911.95	**0.691** / 110.101 / 10,993.11	0.536 / **3.279** / 3,397.83	0.699 / 55.487 / 6,733.52
4. GoT	0.429 / 36.450 / 1,755.87	0.379 / 64.682 / 2,567.15	0.242 / 152.670 / 7,535.46	0.279 / 46.730 / **2,469.44**	0.334 / 653.930 / 4,041.32	0.333 / 190.892 / 3,673.85
5. DPTS	0.656 / **8.781** / 5,471.25	0.734 / 12.734 / 6,501.00	0.013 / **34.470** / 46,422.00	0.436 / **11.287** / 6,060.67	0.260 / 16.109 / 6,341.25	0.420 / **16.676** / 14,159.23
6. SSDP	0.582 / 9.065 / 4,152.38	0.670 / 14.250 / 4,587.08	0.003 / 40.428 / 31,063.00	0.424 / 12.539 / 5,546.73	0.183 / 19.236 / 4,969.68	0.372 / 19.104 / 10,063.77
IToT (Ours) ∗	0.876 / 21.369 / **772.65**	**0.893** / 49.745 / 1,327.92	**0.754** / 50.968 / 3,139.83	0.676 / 55.427 / 3,180.23	**0.584** / 16.984 / 952.94	**0.757** / 38.899 / **1,874.71**
BELEBELE (Accuracy ↑ / Runtime ↓ / Total Token ↓)						
1. CoT	0.858 / 11.040 / **480.16**	0.943 / 25.402 / **881.70**	0.802 / **27.169** / **1,112.82**	0.589 / 94.045 / 3,924.14	0.621 / 9.821 / 1,189.60	0.763 / 33.495 / **1,517.68**
2. CoT-SC	0.779 / 11.040 / **480.16**	0.870 / 25.402 / **881.70**	0.531 / **27.169** / **1,112.82**	0.294 / 94.045 / 3,924.14	0.459 / 9.821 / 1,189.60	0.587 / 33.495 / **1,517.68**
3. ToT	0.863 / 21.554 / 13,054.45	0.884 / **2.401** / 2,266.33	0.717 / 117.020 / 9,885.16	0.580 / 212.244 / 26,235.07	0.553 / **3.315** / 6,441.39	0.719 / 71.307 / 11,576.48
4. GoT	0.754 / 29.340 / 1,800.34	0.888 / 54.470 / 2,505.85	0.623 / 200.320 / 9,662.09	0.431 / 75.600 / **3,838.22**	0.543 / 645.480 / 4,101.22	0.648 / 201.042 / 4,381.54
5. DPTS	0.733 / **7.640** / 7,034.50	0.851 / 12.512 / 7,824.20	0.080 / 33.099 / 43,707.25	0.488 / **13.047** / 8,011.75	0.085 / 14.726 / 6,035.25	0.447 / **16.205** / 14,522.59
6. SSDP	0.605 / 7.994 / 5,359.75	0.857 / 11.493 / 6,071.48	0.063 / 42.389 / 40,258.75	0.458 / 17.358 / 9,032.65	0.148 / 13.333 / 5,783.90	0.426 / 18.513 / 13,301.31
IToT (Ours) ∗	**0.869** / 15.883 / 582.50	**0.949** / 50.522 / 914.01	**0.821** / 47.889 / 1,158.98	**0.608** / 134.740 / 3,978.26	**0.629** / 11.217 / **1,088.73**	**0.775** / 52.050 / 1,544.50
PROSQA (Accuracy ↑ / Runtime ↓ / Total Token ↓)						
1. CoT	0.035 / 77.428 / 3,317.18	0.495 / 116.976 / 3,576.00	0.164 / 99.854 / **3,485.76**	0.040 / 120.460 / 5,061.38	0.322 / **52.153** / 4,849.48	0.211 / 93.374 / 4,057.96
2. CoT-SC	0.012 / 77.428 / 3,317.18	0.315 / 116.976 / 3,576.00	0.062 / 99.854 / **3,485.76**	0.015 / 120.460 / 5,061.38	0.162 / **52.153** / 4,849.48	0.113 / 93.374 / 4,057.96
3. ToT	0.461 / **50.659** / 26,477.57	0.601 / **1.949** / 2,962.52	0.657 / 139.097 / 17,044.25	0.157 / 649.353 / 150,929.21	0.471 / 52.962 / 9,952.33	0.469 / 178.804 / 41,473.18
4. GoT	0.440 / 120.170 / 5,272.56	0.764 / 210.800 / 6,605.61	0.090 / 622.050 / 19,583.60	0.052 / 377.810 / 11,773.58	0.344 / 749.780 / 9,948.14	0.338 / 416.122 / 10,636.70
5. DPTS	0.115 / 82.512 / 154,625.33	0.317 / 53.506 / 58,684.50	0.018 / **45.098** / 30,574.33	0.012 / 60.892 / 65,402.67	0.016 / 64.076 / 24,006.00	0.097 / **61.217** / 66,658.57
6. SSDP	0.051 / 65.406 / 84,789.03	0.190 / 47.789 / 14,434.95	0.007 / 97.784 / 88,244.50	0.013 / **41.269** / 30,638.50	0.022 / 81.769 / 36,815.83	0.057 / 66.803 / 50,984.56
IToT (Ours) ∗	**0.887** / 100.969 / **1,565.22**	**0.943** / 126.703 / **2,551.59**	**0.893** / 60.634 / 3,950.19	**0.503** / 87.220 / **4,921.21**	**0.832** / 74.640 / **4,080.27**	**0.812** / 90.033 / **3,413.70**
MMLU (Accuracy ↑ / Runtime ↓ / Total Token ↓)						
1. CoT	0.704 / 130.203 / 5,348.40	0.835 / 59.847 / 1,845.50	0.754 / 99.764 / 3,442.33	0.614 / 83.528 / 3,529.33	0.431 / **22.442** / **1,559.03**	0.668 / 79.157 / 3,144.92
2. CoT-SC	0.572 / 130.203 / 5,348.40	0.672 / 59.847 / 1,845.50	0.358 / 99.764 / 3,442.33	0.385 / 83.528 / 3,529.33	0.334 / **22.442** / **1,559.03**	0.464 / 79.157 / 3,144.92
3. ToT	**0.776** / 37.676 / 11,487.50	0.725 / **6.941** / **1,795.86**	0.656 / 149.158 / 14,235.76	**0.652** / 143.402 / 23,214.48	0.511 / 30.697 / 7,185.48	0.664 / 73.575 / 11,583.82
4. GoT	0.180 / 76.140 / 2,472.51	0.218 / 122.370 / 3,061.03	0.226 / 235.220 / 8,595.18	0.174 / 96.790 / **3,201.82**	0.008 / 671.570 / 4,503.15	0.161 / 240.418 / 4,366.74
5. DPTS	0.592 / 13.674 / 9,022.40	0.645 / 20.879 / 12,353.25	0.070 / **42.836** / 48,566.33	0.420 / 19.319 / 14,296.00	0.081 / 31.736 / 8,484.67	0.362 / 25.689 / 18,544.53
6. SSDP	0.527 / **11.764** / 6,097.53	0.610 / 17.614 / 7,861.73	0.016 / 43.718 / 39,747.33	0.397 / **17.776** / 10,926.67	0.024 / 24.347 / 7,037.67	0.315 / **23.044** / 14,334.19
IToT (Ours) ∗	0.724 / 20.772 / **1,066.23**	**0.851** / 60.862 / 1,873.98	**0.771** / 60.485 / **3,359.90**	0.633 / 60.825 / 3,509.45	**0.464** / 35.992 / 1,639.37	**0.689** / 47.787 / **2,289.79**

Table 2: Ablation experiments of IToT on GSM8K.

Method	Acc.↑	Tok.↓	Time↓
w/o Steering	0.870	298.78	5.841
w/o Tree	0.859	300.13	**3.777**
w/o Pruning	0.867	299.20	7.454
w/o CMA	0.868	298.91	4.121
w/o Momentum	0.867	300.86	5.378
Single-layer	0.866	**297.92**	5.694
w/o Calibration	0.866	299.22	5.910
IToT (full)	**0.871**	301.01	5.358

momentum updates, the multi-layer design, or offline calibration each reliably lowers accuracy to the 0.866–0.868 range. This demonstrates their collective necessity in refining branch directions and stabilizing the search over extended reasoning horizons. Finally, the results expose a clear efficiency–accuracy trade-off governed by pruning and intervention depth. Pruning is critical for computational efficiency; its removal inflates inference time from 5.358s to 7.454s without improving accuracy. Conversely, restricting the framework to a single-layer variant minimizes token cost (297.92) but at the expense of performance. This implies that while pruning effectively bounds the search space, deep multi-layer iterative steering remains strictly necessary for resolving complex reasoning tasks.

4.4 Parameter Sensitivity

Table 3 details the sensitivity of IToT to four core search hyperparameters on the GSM8K benchmark. Overall, the framework favors moderate control granularity and highly selective search, demonstrating that excessive branching degrades rather than enhances reasoning.

Search Breadth and Retention (B_f and B_w). Contrary to the expectation that wider exploration

Table 3: Sensitivity of IToT on GSM8K. Time is measured in seconds. The default setting is B_f=4, B_w=2, $T_{\max}$=1024, and W=128. * denotes the default value.

Setting	Accuracy ↑	Time/sample ↓
Branch factor (B_f)		
$^*B_f = 4$	0.871	5.358
$B_f = 8$	0.867	5.773
$B_f = 16$	0.864	6.542
Beam width (B_w)		
$B_w = 1$	0.874	4.246
$^*B_w = 2$	0.871	5.358
$B_w = 4$	0.867	7.454
Max tokens ($T_{\max}$)		
$T_{\max} = 512$	0.863	5.249
$^*T_{\max} = 1024$	0.871	5.358
Window size (W)		
$W = 64$	0.861	12.892
$^*W = 128$	0.871	5.358
$W = 256$	0.867	9.237

yields better reasoning, aggressively expanding the branch factor (B_f) or beam width (B_w) proves detrimental. Increasing B_f from 4 to 16 steadily degrades accuracy while inflating inference time. Similarly, restricting the beam width to $B_w = 1$ maximizes accuracy (0.874) and minimizes latency. These combined trends indicate that retaining too many branches introduces noise and unnecessary computational overhead, ultimately diluting search selectivity.

Generation Horizon ($T_{\max}$). Expanding the token budget from 512 to 1024 yields a clear performance improvement (0.863 to 0.871) with only a marginal latency penalty. This confirms that allocating a sufficient generation horizon is strictly beneficial, ensuring the model is not truncated prematurely during complex, multi-step deductions.

Control Granularity (W). IToT exhibits significant sensitivity to the steering window size, revealing a U-shaped efficiency frontier. The default $W = 128$ strikes the optimal balance between accuracy and speed. Deviating from this sweet spot is suboptimal: excessively fine-grained control ($W = 64$) severely bottlenecks inference speed due to frequent CMA-ES updates, while overly coarse intervals fail to provide timely latent guidance. Further hyperparameter analyses are provided in Appendix C.6.

5 Related Work

Tree of Thoughts Reasoning with LLMs. ToT-style reasoning has mainly advanced along two directions: improving search accuracy and reducing computational cost. For accuracy, GoT generalizes tree-structured reasoning to graphs, enabling more flexible reuse and refinement of intermediate thoughts (Besta et al., 2024). MCTS-based methods further improve thought generation, evaluation, and selection, achieving strong performance on mathematical, planning, and logical reasoning tasks (Hao et al., 2023b; Zhang et al., 2024; Guan et al., 2025). For efficiency, prior work reduces latency and token cost through structured generation, cached reasoning templates, fewer model queries, faster decoding, and less unnecessary exploration (Ning et al., 2024; Yang et al., 2024b; Sel et al., 2024; Wang et al., 2025a; Ding et al., 2025; Kim et al., 2025).

Implicit latent space reasoning. Implicit latent space reasoning aims to reduce output length while retaining multi-step reasoning capability. Prior work explores step-by-step internalization of explicit CoT (Deng et al., 2024), latent compression of reasoning traces (Cheng and Durme, 2024; Hao et al., 2024; Shen et al., 2025), and soft thought representations in continuous spaces (Xu et al., 2025; Zhang et al., 2025c; Tan et al., 2026). Other studies support latent reasoning through architectural designs such as looped or adaptive-depth computation (Saunshi et al., 2025; Chen et al., 2025), while recent work further examines its mechanisms and supervision (Wu et al., 2025; Wei et al., 2025). In contrast, our method studies how to simulate ToT-style reasoning directly in latent space without additional training.

6 Conclusion

We presented IToT, a training-free framework that recasts TOT reasoning as continuous search in the latent space of an LLM. Rather than generating and textually scoring candidate branches, IToT searches over latent steering directions guided by a contrastive prototype and injects the selected direction back into the hidden states, unifying branch exploration, evaluation, and execution within a single forward pass. Experiments across a suite of multi-step reasoning benchmarks and several LLM backbones show that IToT consistently improves reasoning quality over strong single-path and explicit tree-search baselines while substantially reducing token consumption, demonstrating that hidden-state geometry is an effective and efficient substrate for structured reasoning.

Limitations

While IToT provides an elegant and highly efficient alternative to explicit text-based reasoning search, it operates under certain boundary conditions that open promising avenues for future research. First, our contrastive prototype is currently estimated from a task-specific calibration set, which may introduce sensitivity to severe domain or prompt shifts. However, this dependency can be easily mitigated by expanding the calibration phase to cross-domain or multi-task datasets, thereby distilling a more universal, task-agnostic "reasoning direction" that enhances out-of-distribution generalization. Second, because IToT optimizes continuous latent trajectories rather than explicit text tokens, its alignment score serves as an indirect surrogate for reasoning quality. To bridge the gap between continuous alignment and discrete logical verification, a hybrid control framework could be deployed in practice, where latent steering guides the broader search space, and explicit textual verifiers are invoked only at a few critical checkpoints or terminal states. Finally, although IToT structurally eradicates the prohibitive token-generation bottleneck of traditional ToT, the continuous search over multiple latent branches inevitably introduces a slight computational overhead compared to vanilla single-trajectory decoding. This engineering constraint can be significantly alleviated via parallelized batch processing of candidate branches or by implementing an adaptive windowing strategy that dynamically skips the search process during low-uncertainty decoding steps.

References

Lucas Bandarkar, Davis Liang, Benjamin Muller, Mikel Artetxe, Satya Narayan Shukla, Donald Husa, Naman Goyal, Abhinandan Krishnan, Luke Zettlemoyer, and Madian Khabsa. 2024. The belebele benchmark: a parallel reading comprehension dataset in 122 language variants. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 749–775.

Maciej Besta, Nils Blach, Ales Kubicek, Robert Gerstenberger, Michal Podstawski, Lukas Gianinazzi, Joanna Gajda, Tomasz Lehmann, Hubert Niewiadomski, Piotr Nyczyk, and 1 others. 2024. Graph of thoughts: Solving elaborate problems with large language models. In *Proceedings of the AAAI conference on artificial intelligence*, volume 38, pages 17682–17690.

Nuo Chen, Ning Wu, Shining Liang, Ming Gong, Linjun Shou, Dongmei Zhang, and Jia Li. 2023. Is bigger and deeper always better? probing llama across scales and layers. *arXiv preprint arXiv:2312.04333*.

Yilong Chen, Junyuan Shang, Zhenyu Zhang, Yanxi Xie, Jiawei Sheng, Tingwen Liu, Shuohuan Wang, Yu Sun, Hua Wu, and Haifeng Wang. 2025. Inner thinking transformer: Leveraging dynamic depth scaling to foster adaptive internal thinking. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 28241–28259, Vienna, Austria. Association for Computational Linguistics.

Jeffrey Cheng and Benjamin Van Durme. 2024. Compressed chain of thought: Efficient reasoning through dense representations. *Preprint*, arXiv:2412.13171.

Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, and 1 others. 2021. Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168*.

Yuntian Deng, Yejin Choi, and Stuart Shieber. 2024. From explicit cot to implicit cot: Learning to internalize cot step by step. *arXiv preprint arXiv:2405.14838*.

Yifu Ding, Wentao Jiang, Shunyu Liu, Yongcheng Jing, Jinyang Guo, Yingjie Wang, Jing Zhang, Zengmao Wang, Ziwei Liu, Bo Du, and 1 others. 2025. Dynamic parallel tree search for efficient llm reasoning. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 11233–11252.

Xuanqi Gao, Haoyu Wang, Jun Sun, Shiqing Ma, and Chao Shen. 2026. Domain-specialized tree of thought through plug-and-play predictors. *arXiv preprint arXiv:2603.20267*.

Xinyu Guan, Li Lyna Zhang, Yifei Liu, Ning Shang, Youran Sun, Yi Zhu, Fan Yang, and Mao Yang. 2025. rstar-math: Small LLMs can master math reasoning with self-evolved deep thinking. In *Forty-second International Conference on Machine Learning*.

Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Peiyi Wang, Qihao Zhu, Runxin Xu, Ruoyu Zhang, Shirong Ma, Xiao Bi, and 1 others. 2025. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. *arXiv preprint arXiv:2501.12948*.

Nikolaus Hansen. 2016. The cma evolution strategy: A tutorial. *arXiv preprint arXiv:1604.00772*.

Shibo Hao, Yi Gu, Haodi Ma, Joshua Hong, Zhen Wang, Daisy Wang, and Zhiting Hu. 2023a. Reasoning with language model is planning with world model. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 8154–8173.

Shibo Hao, Yi Gu, Haodi Ma, Joshua Hong, Zhen Wang, Daisy Wang, and Zhiting Hu. 2023b. Reasoning with language model is planning with world model. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 8154–8173, Singapore. Association for Computational Linguistics.

Shibo Hao, Sainbayar Sukhbaatar, DiJia Su, Xian Li, Zhiting Hu, Jason Weston, and Yuandong Tian. 2024. Training large language models to reason in a continuous latent space. *arXiv preprint arXiv:2412.06769*.

Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt. 2020. Measuring massive multitask language understanding. *arXiv preprint arXiv:2009.03300*.

Dan Hendrycks, Collin Burns, Saurav Kadavath, Akul Arora, Steven Basart, Eric Tang, Dawn Song, and Jacob Steinhardt. 2021. Measuring mathematical problem solving with the math dataset. *arXiv preprint arXiv:2103.03874*.

Yuzhen Huang, Yuzhuo Bai, Zhihao Zhu, Junlei Zhang, Jinghan Zhang, Tangjun Su, Junteng Liu, Chuancheng Lv, Yikai Zhang, Yao Fu, and 1 others. 2023. C-eval: A multi-level multi-discipline chinese evaluation suite for foundation models. *Advances in neural information processing systems*, 36:62991–63010.

Joongho Kim, Xirui Huang, Zarreen Reza, and Gabriel Grand. 2025. Chopping trees: Semantic similarity based dynamic pruning for tree-of-thought reasoning. *arXiv preprint arXiv:2511.08595*.

Hunter Lightman, Vineet Kosaraju, Yuri Burda, Harrison Edwards, Bowen Baker, Teddy Lee, Jan Leike, John Schulman, Ilya Sutskever, and Karl Cobbe. 2023. Let's verify step by step. In *The twelfth international conference on learning representations*.

Nelson F. Liu, Matt Gardner, Yonatan Belinkov, Matthew E. Peters, and Noah A. Smith. 2019. Linguistic knowledge and transferability of contextual representations. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 1073–1094, Minneapolis, Minnesota. Association for Computational Linguistics.

Aman Madaan, Niket Tandon, Prakhar Gupta, Skyler Hallinan, Luyu Gao, Sarah Wiegreffe, Uri Alon, Nouha Dziri, Shrimai Prabhumoye, Yiming Yang, and 1 others. 2023. Self-refine: Iterative refinement with self-feedback. *Advances in neural information processing systems*, 36:46534–46594.

Xuefei Ning, Zinan Lin, Zixuan Zhou, Zifu Wang, Huazhong Yang, and Yu Wang. 2024. Skeleton-of-thought: Prompting llms for efficient parallel generation. In *International Conference on Learning Representations*, volume 2024, pages 917–967.

Aske Plaat, Annie Wong, Suzan Verberne, Joost Broekens, Niki Van Stein, and Thomas Bäck. 2025. Multi-step reasoning with large language models, a survey. *ACM Computing Surveys*, 58(6):1–35.

Qwen, :, An Yang, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chengyuan Li, Dayiheng Liu, Fei Huang, Haoran Wei, Huan Lin, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Yang, Jiaxi Yang, Jingren Zhou, and 25 others. 2025. Qwen2.5 technical report. *Preprint*, arXiv:2412.15115.

Nikunj Saunshi, Nishanth Dikkala, Zhiyuan Li, Sanjiv Kumar, and Sashank J. Reddi. 2025. Reasoning with latent thoughts: On the power of looped transformers. In *International Conference on Learning Representations*, volume 2025, pages 14855–14881.

Bilgehan Sel, Ahmad Tawaha, Vanshaj Khattar, Ruoxi Jia, and Ming Jin. 2024. Algorithm of thoughts: Enhancing exploration of ideas in large language models. In *International Conference on Machine Learning*, pages 44136–44189. PMLR.

Zhenyi Shen, Hanqi Yan, Linhai Zhang, Zhanghao Hu, Yali Du, and Yulan He. 2025. Codi: Compressing chain-of-thought into continuous space via self-distillation. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pages 677–693.

Oscar Skean, Md Rifat Arefin, Dan Zhao, Niket Nikul Patel, Jalal Naghiyev, Yann Lecun, and Ravid Shwartz-Ziv. 2025. Layer by layer: Uncovering hidden representations in language models. In *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pages 55854–55875. PMLR.

Lihao Sun, Hang Dong, Bo Qiao, Qingwei Lin, Dongmei Zhang, and Saravan Rajmohan. 2026. Llm reasoning as trajectories: Step-specific representation geometry and correctness signals. *arXiv preprint arXiv:2604.05655*.

Alon Talmor, Jonathan Herzig, Nicholas Lourie, and Jonathan Berant. 2019. Commonsenseqa: A question answering challenge targeting commonsense knowledge. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4149–4158.

Wenhui Tan, Jiaze Li, Jianzhong Ju, Zhenbo Luo, Ruihua Song, and Jian Luan. 2026. Think silently, think fast: Dynamic latent compression of llm reasoning chains. *Preprint*, arXiv:2505.16552.

Gemma Team, Aishwarya Kamath, Johan Ferret, Shreya Pathak, Nino Vieillard, Ramona Merhej, Sarah Perrin, Tatiana Matejovicova, Alexandre Ramé, Morgane Rivière, Louis Rouillard, Thomas Mesnard, Geoffrey Cideron, Jean bastien Grill, Sabela Ramos, Edouard

Yvinec, Michelle Casbon, Etienne Pot, Ivo Penchev, and 197 others. 2025. Gemma 3 technical report. *Preprint*, arXiv:2503.19786.

Martina G Vilas, Safoora Yousefi, Besmira Nushi, Eric Horvitz, and Vidhisha Balachandran. 2025. Tracing the traces: Latent temporal signals for efficient and accurate reasoning. *arXiv preprint arXiv:2510.10494*.

Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc Le, Ed Chi, Sharan Narang, Aakanksha Chowdhery, and Denny Zhou. 2022. Self-consistency improves chain of thought reasoning in language models. *arXiv preprint arXiv:2203.11171*.

Yiming Wang, Pei Zhang, Baosong Yang, Derek F Wong, and Rui Wang. 2024. Latent space chain-of-embedding enables output-free llm self-evaluation. *arXiv preprint arXiv:2410.13640*.

Zhenglin Wang, Jialong Wu, Yilong Lai, Congzhi Zhang, and Deyu Zhou. 2025a. Seed: Accelerating reasoning tree construction via scheduled speculative decoding. In *Proceedings of the 31st International Conference on Computational Linguistics*, pages 4920–4937.

Zhihai Wang, Jie Wang, Jilai Pan, Xilin Xia, Huiling Zhen, Mingxuan Yuan, Jianye Hao, and Feng Wu. 2025b. Accelerating large language model reasoning via speculative search. In *International Conference on Machine Learning*, pages 64773–64805. PMLR.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Fei Xia, Ed Chi, Quoc V Le, Denny Zhou, and 1 others. 2022. Chain-of-thought prompting elicits reasoning in large language models. *Advances in neural information processing systems*, 35:24824–24837.

Xilin Wei, Xiaoran Liu, Yuhang Zang, Xiaoyi Dong, Yuhang Cao, Jiaqi Wang, Xipeng Qiu, and Dahua Lin. 2025. Sim-cot: Supervised implicit chain-of-thought. *Preprint*, arXiv:2509.20317.

Junhong Wu, Jinliang Lu, Zixuan Ren, Gangqiang Hu, Zhi Wu, Dai Dai, and Hua Wu. 2025. Llms are single-threaded reasoners: Demystifying the working mechanism of soft thinking. *Preprint*, arXiv:2508.03440.

Yige Xu, Xu Guo, Zhiwei Zeng, and Chunyan Miao. 2025. SoftCoT: Soft chain-of-thought for efficient reasoning with LLMs. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 23336–23351, Vienna, Austria. Association for Computational Linguistics.

An Yang, Beichen Zhang, Binyuan Hui, Bofei Gao, Bowen Yu, Chengpeng Li, Dayiheng Liu, Jianhong Tu, Jingren Zhou, Junyang Lin, Keming Lu, Mingfeng Xue, Runji Lin, Tianyu Liu, Xingzhang Ren, and Zhenru Zhang. 2024a. Qwen2.5-math technical report: Toward mathematical expert model via self-improvement. *Preprint*, arXiv:2409.12122.

Ling Yang, Zhaochen Yu, Tianjun Zhang, Shiyi Cao, Minkai Xu, Wentao Zhang, Joseph E Gonzalez, and Bin Cui. 2024b. Buffer of thoughts: Thought-augmented reasoning with large language models. *Advances in Neural Information Processing Systems*, 37:113519–113544.

Shunyu Yao, Dian Yu, Jeffrey Zhao, Izhak Shafran, Tom Griffiths, Yuan Cao, and Karthik Narasimhan. 2023. Tree of thoughts: Deliberate problem solving with large language models. *Advances in neural information processing systems*, 36:11809–11822.

Eric Zelikman, Georges Harik, Yijia Shao, Varuna Jayasiri, Nick Haber, and Noah D Goodman. 2024. Quiet-star: Language models can teach themselves to think before speaking. *arXiv preprint arXiv:2403.09629*.

Anqi Zhang, Yulin Chen, Jane Pan, Chen Zhao, Aurojit Panda, Jinyang Li, and He He. 2025a. Reasoning models know when they're right: Probing hidden states for self-verification. *arXiv preprint arXiv:2504.05419*.

Dan Zhang, Sining Zhoubian, Ziniu Hu, Yisong Yue, Yuxiao Dong, and Jie Tang. 2024. Rest-mcts$*$: $Llmself-trainingviaprocessrewardguidedtreesearch$. In *Advances in Neural Information Processing Systems*, volume 37, pages 64735–64772. Curran Associates, Inc.

Zhen Zhang, Xuehai He, Weixiang Yan, Ao Shen, Chenyang Zhao, Shuohang Wang, Yelong Shen, and Xin Eric Wang. 2025b. Soft thinking: Unlocking the reasoning potential of llms in continuous concept space. *arXiv preprint arXiv:2505.15778*.

Zhen Zhang, Xuehai He, Weixiang Yan, Ao Shen, Chenyang Zhao, Shuohang Wang, Yelong Shen, and Xin Eric Wang. 2025c. Soft thinking: Unlocking the reasoning potential of llms in continuous concept space. *Preprint*, arXiv:2505.15778.

Lianmin Zheng, Wei-Lin Chiang, Ying Sheng, Siyuan Zhuang, Zhanghao Wu, Yonghao Zhuang, Zi Lin, Zhuohan Li, Dacheng Li, Eric Xing, and 1 others. 2023. Judging llm-as-a-judge with mt-bench and chatbot arena. *Advances in neural information processing systems*, 36:46595–46623.

Appendix

A Preliminaries

To operationalize this latent-space guidance, we first formalize the representation space where we extract quality signals and intervene. Consider a pretrained decoder-only LLM parameterized by θ with L Transformer blocks, represented as:

$$f_\theta = f_\theta^{(L)} \circ \cdots \circ f_\theta^{(1)}.$$

Given an input x and an autoregressively generated response $y = (y_1, \ldots, y_T)$, the hidden state at decoding step t and layer l is defined as:

$$\mathbf{h}_t^{(l)} = f_\theta^{(l)}\left(\mathbf{h}_t^{(l-1)}; x, y_{<t}\right).$$

Based on our observation regarding upper-layer abstraction, we restrict our analysis and intervention to the upper layers, denoted by:

$$\mathcal{L}_{\text{high}} = \{l_{\text{start}}, \ldots, L\}.$$

Because our objective is to influence high-level reasoning rather than surface token realization, these upper-layer states provide the optimal intervention space.To construct a reference point for branch evaluation, we rely on offline calibration using completed traces. We summarize a full reasoning trace by its final-token upper-layer states:

$$\mathbf{z}^{(l)}(x, y) = \mathbf{h}_T^{(l)}, \qquad l \in \mathcal{L}_{\text{high}}.$$

These terminal states act as response-level latent representations. Conversely, during online inference, our controller operates dynamically on the upper-layer states at the end of each decoding window, denoted by $\mathbf{h}_{t_w}^{(l)}$, where t_w is the window's final token. Thus, the same upper-layer manifold seamlessly supports both offline calibration and online branch control.Contrastive Latent Prototype. To distinguish good reasoning from bad within this space, we leverage a calibration set:

$$\mathcal{D}_{\text{cal}} = \{(x_i, y_i^+, y_i^-)\}_{i=1}^N,$$

where y_i^+ and y_i^- are successful and failed traces for the same input x_i. Since these traces diverge systematically in representation space (Wang et al., 2024; Vilas et al., 2025), we construct a contrastive prototype by averaging the direction from failure to success across all calibration instances. For each $l \in \mathcal{L}_{\text{high}}$, we define:

$$\mathbf{p}^{(l)} = \text{Norm}\left(\frac{1}{N}\sum_{i=1}^{N}\left(\mathbf{z}^{(l)}(x_i, y_i^+) - \mathbf{z}^{(l)}(x_i, y_i^-)\right)\right),$$

where $\text{Norm}(\cdot)$ is ℓ_2 normalization. This prototype $\mathbf{p}^{(l)}$ captures the average latent trajectory toward successful reasoning. We treat this prototype not as a flawless verifier, but as a highly efficient quality proxy; its alignment with current hidden states serves as our evaluation metric, circumventing the need for explicit text decoding.

B Pseudo code of IToT

Algorithm 1 summarizes the IToT method.

C Experiment

C.1 Datasets

We select six datasets across four domains in our main experiments. We describe each dataset's information as follows:

- *Mathematics Domain*
  - **GSM8K** (Cobbe et al., 2021) consists of linguistically diverse math word problems specifically designed for grade school students and created by human writers. It contains 1319 test problems, each requiring 2-8 steps to solve. The challenges primarily involve a sequence of basic arithmetic operations (addition, subtraction, multiplication, and division) to derive the final answer.
  - **MATH** (Hendrycks et al., 2021) consists of challenging competition mathematics problems. It contains 500 test problems from diverse domains including algebra, geometry, counting and probability, number theory, and precalculus.
- *Reasoning Domain*
  - **CommonsenseQA** (Talmor et al., 2019) consists of diverse commonsense question answering questions. It contains 1221 test problems, each with 5 choices. This dataset was generated by Amazon Mechanical Turk workers.
  - **PROSQA** (Hao et al., 2024) is a logical reasoning benchmark designed to evaluate multi-step reasoning and search capabilities. Each instance is represented as a directed acyclic graph (DAG) of natural language statements, where nodes denote concepts or facts and edges represent logical relations. Models must identify valid reasoning paths through the graph to answer questions correctly.
- *Knowledge Domain*
  - **MMLU** (Hendrycks et al., 2020) consists of diverse questions that measure knowledge acquired during pretraining. This dataset covers 57 subjects across STEM, the humanities, the social sciences, and more. It ranges in difficulty from an elementary level to an advanced

Table 4: Summary of evaluated datasets and their sizes.

Dataset	Number of Samples
GSM8K	1,319
MATH	500
CommQA	1,000
Belebele	900
ProSQA	500
MMLU	1,000

professional level, can test both world knowledge and problem-solving ability.

- *Understanding Domain*
- **Belebele** (Bandarkar et al., 2024) consists of multilingual, multitask, and multidisciplinary reading comprehension questions, with each language containing 900 article-question pairs. We use the English version of this dataset.

The statistics of the evaluated datasets are summarized in Table 4.

C.2 Language Models

These models span general-purpose and math-specialized instruction-following variants, enabling us to assess robustness across architectures and training objectives. All models are evaluated under a zero-shot inference setting without task-specific fine-tuning. All model checkpoints were downloaded from the official HuggingFace[1] repository, with download and license links provided in Table 5.

C.3 Baselines

We describe each baseline's information as follows:

- **CoT** (Wei et al., 2022) Chain of Thought prompting induces intermediate reasoning steps in a single linear generation process. Given an input query, the model is prompted to produce a step-by-step rationale before outputting the final answer. This approach does not involve explicit search or sampling and relies on a single deterministic or stochastic decoding trajectory.

- **CoT-SC** (Wang et al., 2022) Self-Consistency extends CoT by sampling multiple independent reasoning chains via stochastic decoding. Final answers are obtained by aggregating the outputs, typically through majority voting. This improves robustness by mitigating variance in single-chain reasoning.

- **ToT** (Yao et al., 2023) Tree of Thoughts formulates reasoning as a tree-structured search problem, where each intermediate step can branch into multiple candidate thoughts. The model performs systematic exploration of the reasoning space using search strategies such as breadth-first search or depth-first search, combined with heuristic evaluation and pruning.

- **GoT** (Besta et al., 2024) Graph of Thoughts generalizes tree-based reasoning into a graph structure, allowing non-linear transitions between reasoning states. Nodes represent intermediate thoughts and edges encode transitions or transformations, enabling reuse, merging, and revisiting of reasoning components during search.

- **DPTS** (Ding et al., 2025) DPTS introduces dynamic parallelism into tree-based reasoning. It adaptively allocates computation across branches according to intermediate scoring signals, enabling selective expansion and pruning of the search tree. The method improves efficiency by concentrating computation on high-value reasoning paths.

- **SSDP** (Kim et al., 2025) SSDP builds upon DPTS by incorporating semantic evaluation into the reasoning search process. It scores intermediate states using semantic relevance signals and performs dynamic pruning of low-quality branches during exploration. This results in a more computationally efficient search over the reasoning space while preserving high-quality reasoning paths.

These baselines span representative explicit reasoning paradigms, ranging from linear chain-based prompting (CoT) and multi-path sampling with majority voting (CoT-SC), to tree- and graph-structured search (ToT, GoT), and efficiency-oriented structured search with dynamic parallelism and semantic pruning (DPTS, SSDP).

[1] https://huggingface.co/huggingface

Table 5: Download and license links of all model artifacts we used in this paper.

Model Name	Download Link	License Link
Qwen2.5-1.5B-Instruct	https://huggingface.co/Qwen/Qwen2.5-1.5B-Instruct	https://huggingface.co/Qwen/Qwen2.5-1.5B-Instruct/blob/main/LICENSE
Qwen2.5-3B-Instruct	https://huggingface.co/Qwen/Qwen2.5-3B-Instruct	https://huggingface.co/Qwen/Qwen2.5-3B-Instruct/blob/main/LICENSE
DeepSeek-R1-Distill-Qwen-1.5B	https://huggingface.co/deepseek-ai/DeepSeek-R1-Distill-Qwen-1.5B	https://huggingface.co/deepseek-ai/DeepSeek-R1-Distill-Qwen-1.5B/blob/main/LICENSE
Qwen2.5-Math-1.5B-Instruct	https://huggingface.co/Qwen/Qwen2.5-Math-1.5B-Instruct	https://huggingface.co/Qwen/Qwen2.5-Math-1.5B-Instruct/blob/main/LICENSE
Gemma-3-1B-IT	https://huggingface.co/google/gemma-3-1b-it	https://ai.google.dev/gemma/terms

Table 6: Hyperparameters of Implicit Tree of Thoughts via Latent-Space Beam Steering.

Parameter	Symbol	Default
Max new tokens	$T_{\max}$	1024
Sampling temperature	τ	0.7
Number of calibration pairs	N_{cal}	10
Number of branches	B_f	4
Beam width	B_w	2
Token window for update	W	128
Momentum decay coefficient	β	0.9
Population size	λ	8
Number of generations	G	4
Initial direction noise	σ_0	0.03
Mutation noise	σ_m	0.01
Soft mixture weight	α	0.5

C.4 Implementation Details

To ensure a fair comparison, all methods use the same backbone LLM and decoding settings unless otherwise specified. For the baseline CoT prompting, the sampling temperature is set to 0.7, and the maximum generation length is limited to 1024 tokens. The key hyperparameters of our method are summarized in Table 6. Time is measured in seconds.

For ToT, we adopt a breadth-first search strategy. At each reasoning step, the model generates $k = 4$ candidate thoughts, while the top $b = 3$ states are retained according to the evaluation score. The search process proceeds for at most 6 reasoning steps, and the final top-5 candidate answers are returned for answer selection. For each thought generation step, the sampling temperature is set to 0.7, and the maximum generation length is limited to 1024 tokens.

For GoT , the number of reasoning branches is set to 3, and one refinement round is performed during reasoning.

For both DPTS and SSDP, the search tree width is set to 4 and the maximum tree depth is set to 6. In addition, the maximum rollout number is fixed at 6, and the maximum generation length for each rollout is limited to 1024 tokens.

All remaining hyperparameters are kept at their default settings.

Unless otherwise specified, all evaluations are conducted on five candidate responses generated within a single run.

C.5 Implicit Reasoning

We describe each implicit reasoning baseline's information as follows:

- **Coconut** (Hao et al., 2024) Coconut represents a latent or implicit reasoning paradigm in which intermediate reasoning steps are not explicitly generated in natural language. Instead, reasoning is conducted in a compressed latent space, reducing token-level supervision of intermediate steps.

- **CODI** (Shen et al., 2025) CODI is a distillation-based method where explicit CoT acts as the teacher andimplicit CoT as the student. By aligning the last hidden states of the full reasoning trajectory, CODI effectively internalizes knowledge and alleviates catastrophic forgetting.

Implementation Details. We report the implementation details for Coconut and CODI. All remaining hyperparameters not explicitly mentioned are kept at their default settings.

For Coconut, we set the number of continuous thoughts to 2, the number of epochs per stage to 3, and the maximum latent stage to 3. The training batch size is 8 with gradient accumulation steps set to 1. The model is trained for 25 epochs with a learning rate of 1×10^{-4} and a weight decay of 0.01.

For CODI, we adopt a LoRA-based training setup with rank $r = 128$, LoRA dropout of 0.05, and scaling factor $\alpha = 16$. The model is trained for 3 epochs with per-device batch size 2 and gradient accumulation steps of 1. We use the AdamW optimizer with a learning rate of 2×10^{-5}, weight decay of 0.01, warmup ratio of 0.1, and cosine learning rate scheduling. The number of latent variables is set to 5, and the maximum sequence length is 1024 tokens. The implementation further uses bf16 precision and sets the distillation loss to smooth L1 with equal weighting for reference and distillation losses.

Unlike the main experiments, this evaluation is conducted using only a single candidate response generated from one run.

Table 7: Implicit reasoning results on accuracy and efficiency across six datasets. Each entry reports accuracy ↑, runtime per sample ↓, and total token consumption per sample ↓. Accuracy and runtime are reported to three decimal places, while token consumption is reported to two decimal places. * denotes results reported in the main text. Δ denotes evaluation using the COMMONSENSEQA-trained weights.

Method / Model	Qwen2.5-1.5B	Qwen2.5-3B	DeepSeek-R1-Distill-Qwen-1.5B	Qwen2.5-Math-1.5B	Gemma-3-1B	Avg.
GSM8K (Accuracy ↑ / Runtime ↓ / Total Token ↓)						
1. Coconut	0.039 / 0.299 / 4.54	0.085 / 0.305 / 4.49	0.086 / 1.116 / 25.33	0.071 / 0.310 / 4.54	0.027 / 0.390 / 3.84	0.062 / 0.484 / 8.55
2. CODI	0.002 / 3.195 / 73.80	0.120 / 0.894 / 11.10	0.113 / 0.715 / 11.10	0.206 / 0.592 / 8.50	0.024 / 11.481 / 234.30	0.093 / 3.375 / 67.76
IToT(P@1) (Ours) *	0.669 / 5.358 / 301.01	0.802 / 9.971 / 299.96	0.489 / 15.820 / 928.34	0.765 / 6.141 / 353.77	0.637 / 13.313 / 627.37	0.672 / 10.121 / 502.09
MATH500 (Accuracy ↑ / Runtime ↓ / Total Token ↓)						
1. Coconut	0.038 / 0.297 / 4.33	0.080 / 0.315 / 4.58	0.044 / 3.869 / 95.18	0.076 / 0.332 / 4.93	0.028 / 0.367 / 2.74	0.053 / 1.036 / 22.35
2. CODI	0.116 / 0.651 / 9.90	0.068 / 2.122 / 35.00	0.018 / 1.182 / 23.20	0.164 / 0.634 / 9.60	0.010 / 4.100 / 80.30	0.075 / 1.738 / 31.60
IToT(P@1) (Ours) *	0.476 / 9.532 / 542.87	0.626 / 18.924 / 567.62	0.196 / 16.809 / 986.90	0.648 / 9.698 / 558.71	0.416 / 14.809 / 704.25	0.472 / 13.954 / 672.07
COMMONSENSEQA (Accuracy ↑ / Runtime ↓ / Total Token ↓)						
1. Coconut	0.008 / 0.265 / 2.78	0.032 / 0.256 / 2.70	0.001 / 0.338 / 2.90	0.023 / 0.256 / 2.66	0.038 / 1.106 / 25.04	0.020 / 0.444 / 7.22
2. CODI	0.193 / 0.569 / 7.20	0.193 / 1.820 / 27.60	0.192 / 0.543 / 7.0	0.193 / 0.575 / 7.50	0.197 / 4.943 / 97.20	0.194 / 1.690 / 29.30
IToT(P@1) (Ours) *	0.673 / 3.848 / 154.53	0.627 / 9.086 / 265.58	0.378 / 10.657 / 627.97	0.332 / 10.791 / 636.05	0.448 / 4.158 / 190.59	0.492 / 7.708 / 374.94
BELEBELE (Accuracy ↑ / Runtime ↓ / Total Token ↓)						
1. Coconut Δ	0 / 0.258 / 2.77	0 / 0.270 / 2.82	0 / 0.344 / 2.91	0 / 0.262 / 2.71	0.003 / 1.154 / 25.18	0.001 / 0.458 / 7.28
2. CODI Δ	0.184 / 0.718 / 10.40	0.102 / 7.294 / 127.60	0.002 / 0.622 / 8.20	0.093 / 0.774 / 11.80	0.009 / 7.474 / 143.40	0.078 / 3.376 / 60.28
IToT(P@1) (Ours) *	0.759 / 3.710 / 114.87	0.859 / 10.224 / 182.76	0.489 / 8.106 / 231.23	0.328 / 26.947 / 796.82	0.449 / 2.102 / 220.10	0.577 / 10.218 / 309.16
PROSQA (Accuracy ↑ / Runtime ↓ / Total Token ↓)						
1. Coconut	0.728 / 0.404 / 7.66	0.604 / 0.414 / 7.66	0.012 / 6.534 / 125.20	0.688 / 0.420 / 7.62	0.064 / 1.337 / 29.68	0.419 / 1.822 / 35.56
2. CODI	0.648 / 0.772 / 11.70	0.730 / 3.564 / 57.60	0.334 / 2.467 / 53.20	0.554 / 0.777 / 11.70	0.006 / 0.840 / 10.30	0.454 / 1.684 / 28.90
IToT(P@1) (Ours) *	0.487 / 21.758 / 363.31	0.572 / 25.844 / 509.67	0.473 / 13.308 / 738.875	0.276 / 17.4175 / 983.65	0.492 / 14.928 / 734.29	0.460 / 18.651 / 665.96
MMLU (Accuracy ↑ / Runtime ↓ / Total Token ↓)						
1. Coconut	0.156 / 0.348 / 5.67	0.203 / 0.403 / 7.80	0.151 / 0.613 / 10.35	0.183 / 0.452 / 9.59	0 / 0.673 / 11.90	0.139 / 0.498 / 9.06
2. CODI	0.007 / 0.899 / 14.40	0.054 / 2.246 / 34.10	0.095 / 9.586 / 170.00	0.003 / 1.183 / 21.40	0.003 / 8.494 / 169.10	0.032 / 4.482 / 81.80
IToT(P@1) (Ours) *	0.581 / 4.154 / 213.48	0.666 / 11.856 / 374.79	0.372 / 12.097 / 671.22	0.381 / 12.165 / 693.82	0.337 / 7.066 / 327.87	0.467 / 9.468 / 456.24

Analysis. Table 7 reveals a clear trade-off between lightweight implicit reasoning baselines and our proposed IToT. Existing methods, including Coconut and CODI, show notable strengths in efficiency. In particular, Coconut is consistently the fastest method across nearly all datasets and model backbones, while also using the fewest tokens per sample. This suggests that Coconut is well-suited for highly budget-constrained settings where low latency and minimal decoding cost are prioritized over raw task performance. CODI, although generally more expensive than Coconut, still remains substantially more efficient than IToTin both runtime and token consumption, and on some tasks it offers modest accuracy gains over Coconut. These results indicate that prior implicit reasoning approaches provide a favorable efficiency profile and can serve as practical solutions when computational resources are limited.

At the same time, our method demonstrates a fundamentally different advantage: substantially stronger reasoning effectiveness. Across GSM8K, MATH500, COMMONSENSEQA, BELEBELE, and MMLU, IToTachieves the best average accuracy by a large margin, often improving over both Coconut and CODI by a wide absolute gap. This trend is especially pronounced on reasoning-intensive benchmarks such as GSM8K, MATH500, and MMLU, where IToTconsistently delivers much higher accuracy across most backbones. The gains remain robust even under heterogeneous model families, including Qwen, DeepSeek-Distill, and Gemma, suggesting that the benefit of IToTis not tied to a single architecture. These results support the view that IToTallocates additional computation in a way that translates into meaningfully better latent reasoning quality, rather than merely increasing cost.

An interesting exception appears on PROSQA, where Coconut and CODI remain competitive and in several cases outperform IToT. This suggests that for tasks with stronger retrieval, shortcut, or shallow pattern components, extremely lightweight implicit reasoning may already be sufficient, and the additional reasoning depth introduced by IToTdoes not always yield proportional gains. Nevertheless, considering all six datasets together, IToTprovides the strongest overall accuracy and the most consistent reasoning improvements across diverse tasks, making it the most effective method when final performance is the primary objective.

C.6 Parameter sensitivity

In §4.4, we present preliminary results on selected hyperparameters. In this section, we conduct a comprehensive analysis of all hyperparameters listed in Table 6.

Table 8: Joint sensitivity of IToTon GSM8K. Each entry reports P@5↑ / Time↓ (s). We fix $T_{\max} = 1024$. * denotes results reported in the main text.

W	$B_f \backslash B_w$	1	2	4	8	16
64	4	0.858/10.515	$^{*}0.861/12.892$	0.864/17.368	–	–
	8	0.867/11.077	0.868/13.154	0.869/17.728	0.873/26.663	–
	16	0.876/11.737	0.870/14.002	0.877/18.334	0.860/27.550	0.866/45.629
128	4	$^{*}0.874/4.246$	$^{*}0.871/5.358$	$^{*}0.867/7.454$	–	–
	8	0.886/4.615	$^{*}0.867/5.773$	0.881/8.005	0.870/12.587	–
	16	0.872/5.429	$^{*}0.864/6.542$	0.868/9.341	0.875/13.527	0.859/24.205
256	4	0.868/8.743	$^{*}0.867/9.237$	0.872/10.446	–	–
	8	0.865/9.165	0.860/9.708	0.876/10.619	0.883/13.032	–
	16	0.876/10.225	0.862/10.789	0.876/11.797	0.861/13.595	0.864/17.757

Joint sensitivity in branches and beam width. Table 8 reports the joint sensitivity of IToTto the token window size W, the number of branches B_f, and the beam width B_w on GSM8K, while fixing $T_{\max} = 1024$. Overall, IToTis fairly robust in accuracy across a broad range of settings: P@5 stays within a relatively narrow band (0.858 to 0.886), whereas inference time varies much more substantially (4.246s to 45.629s). This suggests that the main effect of larger search budgets is increased computational cost rather than consistent accuracy gains.

Among the tested window sizes, $W = 128$ provides the best overall efficiency-accuracy trade-off. It yields not only the fastest configurations in the table, but also the best observed P@5 of 0.886 at $(W, B_f, B_w) = (128, 8, 1)$. By contrast, $W = 64$ is consistently slower and only occasionally improves accuracy, while $W = 256$ does not provide clear accuracy benefits over $W = 128$ despite noticeably higher latency in many settings. This indicates that a moderate update window is sufficient to capture useful latent steering signals on GSM8K.

Increasing B_w or B_f generally enlarges the search budget and therefore increases runtime, but the accuracy improvements are not monotonic. For example, under $W = 128$, moving from $B_w = 1$ to larger beam widths often increases latency with little or even negative impact on P@5. Similarly, larger B_f can help in some cases, but the benefit is modest and inconsistent relative to the added cost. These results support our default choice of $(W, B_f, B_w) = (128, 4, 2)$ as a balanced operating point: it achieves competitive accuracy (0.871) with low runtime (5.358s), while avoiding the steep cost increase of wider search configurations.

Effect of Decoding Budget under Different Search Widths. Table 9 compares the joint sensitivity of IToTto B_f, B_w, and the decoding budget $T_{\max} \in 512, 1024$ under a fixed update window $W = 128$. Overall, increasing $T_{\max}$ from 512 to 1024 does not uniformly improve performance, but it tends to provide slightly better peak accuracy at a modest runtime cost. In particular, the best result in the table is obtained at $(T_{\max}, B_f, B_w) = (1024, 8, 1)$ with P@5 = 0.886, compared with the best 512-token setting of 0.881 at $(512, 8, 2)$.

A Larger Token Budget Mainly Improves the Upper Envelope Rather Than Every Configuration. For many matched settings, the gain from $T_{\max} = 1024$ is small and sometimes inconsistent. For example, with $B_f = 4$, the 1024-token setting is slightly better than the 512-token setting across $B_w \in 1, 2, 4$, but the margins are limited. Under $B_f = 8$ and $B_f = 16$, some configurations improve while others remain similar or even fluctuate slightly downward, suggesting that a larger decoding budget mostly helps preserve headroom for strong search settings rather than delivering uniform gains across the board.

Runtime Growth Remains Moderate Compared with Search Expansion. Moving from $T_{\max} = 512$ to 1024 consistently increases inference time, but the increase is much smaller than the cost induced by enlarging B_f or B_w. This indicates that, in this regime, search width is the dominant factor controlling latency, while the decoding budget plays a secondary role. Taken together, these results support using $T_{\max} = 1024$ when peak performance is preferred, while $T_{\max} = 512$ remains a competitive and more economical option for efficiency-oriented

Table 9: Joint sensitivity of IToTon GSM8K. Each entry reports P@5↑ / Time↓ (s). We compare $T_{\max} \in \{512, 1024\}$. * denotes results reported in the main text.

$B_f \backslash B_w$	1	2	4	8	16
			$T_{\max} = 512$		
4	0.870/4.168	0.863/5.249	0.857/7.424	–	–
8	0.851/4.594	0.881/5.607	0.864/7.768	0.872/12.043	–
16	0.866/5.441	0.862/6.381	0.858/8.546	0.862/12.876	0.870/22.026
			$T_{\max} = 1024$		
4	*0.874/4.246	*0.871/5.358	*0.867/7.454	–	–
8	0.886/4.615	*0.867/5.773	0.881/8.005	0.870/12.587	–
16	0.872/5.429	*0.864/6.542	0.868/9.341	0.875/13.527	0.859/24.205

Table 10: Sensitivity on GSM8K. Time is measured in seconds. The default setting is $T_{\max}$=1024, W=128, and G=4. * denotes the default value.

Setting	Acc. ↑	Time/sample ↓
Generations (G)		
$G = 1$	0.867	4.511
$G = 2$	0.861	4.791
$^*G = 4$	0.871	5.358
$G = 8$	0.868	6.267
$G = 16$	0.855	8.359

Table 11: Sensitivity on GSM8K. Time is measured in seconds. The default setting is $T_{\max}$=1024, W=128, and λ=8. * denotes the default value.

Setting	Acc. ↑	Time/sample ↓
Population size (λ)		
$\lambda = 4$	0.861	5.183
$^*\lambda = 8$	0.871	5.358
$\lambda = 16$	0.873	5.433

settings.

Sensitivity to the Number of Generations. Table 10 shows that IToT is moderately sensitive to the evolutionary search depth G, with the default setting $G = 4$ achieving the best overall accuracy. In particular, $G = 4$ attains the highest P@5 of 0.871, outperforming both shallower and deeper search settings.

A Moderate Number of Generations Is Sufficient for Effective Refinement. When G is reduced to 1 or 2, acc drops slightly to 0.867 and 0.861, suggesting that too few generations may not provide enough iterative refinement for the latent steering directions. On the other hand, increasing G beyond 4 does not improve performance: acc decreases to 0.868 at $G = 8$ and further to 0.855 at $G = 16$. This indicates that deeper evolutionary search brings diminishing returns and may even harm optimization quality.

Runtime Increases Gradually with Larger G. Unlike the earlier setting where runtime differences were more dramatic, the updated results show a smoother growth in time from 4.511s at $G = 1$ to 8.359s at $G = 16$. Even so, the extra cost of larger G is not matched by better accuracy. Taken together, these results support the default choice $G = 4$ as the best balance between search effectiveness and computational efficiency.

Sensitivity to Population Size. Table 11 shows that IToT is highly stable with respect to the population size λ. Across the tested values, acc varies only slightly from 0.861 to 0.873, while runtime remains nearly unchanged, indicating that the method is not particularly sensitive to this hyperparameter in the examined range.

Larger Populations Bring Only Marginal Accuracy Gains. Increasing the population size from $\lambda = 4$ to $\lambda = 16$ improves acc from 0.861 to 0.873, but the gains are gradual and modest. The default setting $\lambda = 8$ already achieves strong performance with acc = 0.871, very close to the best observed result.

The Default Choice Remains a Good Practical Setting. Since runtime differs only slightly across all three settings, the main effect of increasing λ is a small improvement in accuracy rather than a major efficiency trade-off. Taken together, these results suggest that IToT is robust to population size, and

Table 12: Sensitivity on GSM8K. Time is measured in seconds. The default setting is $T_{\max}$=1024, W=128, and σ_0=0.03. * denotes the default value.

Setting	Acc. ↑	Time/sample ↓
Initial noise (σ_0)		
$\sigma_0 = 0.01$	0.869	5.312
$^*\sigma_0 = 0.03$	0.871	5.358
$\sigma_0 = 0.05$	0.864	5.324
$\sigma_0 = 0.1$	0.865	5.301
$\sigma_0 = 0.2$	0.877	5.340
$\sigma_0 = 0.5$	0.864	5.361
$\sigma_0 = 0.9$	0.883	5.271

they support $\lambda = 8$ as a reasonable default that balances performance and search stability.

Sensitivity to Initial Direction Noise. Table 12 shows that IToT is quite robust to the initial noise scale σ_0, since runtime remains almost identical across all tested values and accuracy varies within a relatively narrow range. This indicates that the method is not highly sensitive to the precise initialization magnitude.

Larger Initial Noise Can Sometimes Improve Accuracy, but the Trend Is Not Monotonic. The best acc is achieved at $\sigma_0 = 0.9$ with 0.883, followed by 0.877 at $\sigma_0 = 0.2$, while smaller and intermediate values produce slightly lower results. At the same time, several nearby settings, such as 0.05, 0.1, and 0.5, do not show similar gains, suggesting that the effect of increasing σ_0 is not systematic but depends on how exploration interacts with the downstream search process.

The Default Setting Remains a Stable and Reasonable Choice. Although some larger noise scales achieve somewhat higher acc, the default $\sigma_0 = 0.03$ still provides competitive accuracy at essentially the same runtime. Taken together, these results suggest that IToT tolerates a broad range of initialization strengths, and the default choice remains a sensible operating point without requiring fine-grained tuning.

Sensitivity to Mutation Noise. Table 13 shows that IToT is fairly robust to the mutation noise scale σ_m, as runtime remains nearly constant across all tested values and acc varies within a limited range. This suggests that the method can tolerate different levels of perturbation during evolutionary updates without major changes in computational cost.

Moderate Mutation Noise Appears Most Helpful. The best performance is obtained at $\sigma_m = 0.09$, which achieves acc = 0.878, while $\sigma_m = 0.05$ also

Table 13: Sensitivity on GSM8K. Time is measured in seconds. The default setting is $T_{\max}$=1024, W=128, and σ_m=0.01. * denotes the default value.

Setting	Acc. ↑	Time/sample ↓
Mutation noise (σ_m)		
$^*\sigma_m = 0.01$	0.871	5.358
$\sigma_m = 0.03$	0.860	5.273
$\sigma_m = 0.05$	0.872	5.295
$\sigma_m = 0.09$	0.878	5.307
$\sigma_m = 0.5$	0.863	5.352
$\sigma_m = 0.9$	0.853	5.288

Table 14: Sensitivity on GSM8K. Time is measured in seconds. The default setting is $T_{\max}$=1024, W=128, and β=0.9. * denotes the default value.

Setting	Acc. ↑	Time/sample ↓
Momentum (β)		
$\beta = 0.1$	0.858	5.306
$\beta = 0.3$	0.867	5.348
$\beta = 0.5$	0.878	5.373
$^*\beta = 0.9$	0.871	5.358

slightly outperforms the default with acc = 0.872. In contrast, both very small and very large mutation noise levels are less favorable: the default $\sigma_m = 0.01$ remains competitive at 0.871, but performance drops to 0.863 at $\sigma_m = 0.5$ and 0.853 at $\sigma_m = 0.9$. This pattern suggests that moderate mutation strength improves exploration, whereas overly large perturbations may destabilize the refinement process.

The Default Setting Remains a Reasonable Conservative Choice. Although $\sigma_m = 0.09$ yields the highest acc, the default $\sigma_m = 0.01$ is very close in performance and performs consistently well. Taken together, these results indicate that IToT benefits from some mutation-driven exploration, but does not require aggressive noise, making the default setting a stable and practical choice.

Sensitivity to Momentum Decay. Table 14 shows that IToT is only mildly sensitive to the momentum decay coefficient β, since runtime stays almost constant across all tested values while acc changes within a relatively small range. This indicates that momentum mainly affects optimization quality rather than computational cost.

Intermediate Momentum Strength Appears Most Effective. As β increases from 0.1 to 0.5, acc improves from 0.858 to 0.878, suggesting that retaining more historical update information helps stabilize and refine the latent search direction. However,

Table 15: Sensitivity on GSM8K. Time is measured in seconds. The default setting is N_{cal}=10. * denotes the default value. We report results using both a single candidate response and five candidate responses generated within one run.

Setting	P@1 ↑	P@5 ↑	Time/sample ↓
Calibration pairs (N_{cal})			
$^*N_{cal} = 10$	0.669	0.871	5.358
$N_{cal} = 100$	0.673	0.859	5.644
$N_{cal} = 1000$	0.672	0.873	5.863

the trend is not strictly monotonic, since the default setting $\beta = 0.9$ yields 0.871, which is slightly below the best result at $\beta = 0.5$.

The Default Setting Remains a Stable Practical Choice. Although $\beta = 0.5$ achieves the highest acc, the margin over the default is modest, and all settings have nearly identical runtime. Taken together, these results suggest that IToT is robust to the choice of momentum decay within a broad range, while the default $\beta = 0.9$ remains a reasonable configuration that delivers strong performance without sensitive tuning.

Sensitivity to the Number of Calibration Pairs. Table 15 shows that IToTis quite robust to the number of calibration pairs N_{cal}. Both P@1 and P@5 vary only slightly across the tested settings, while runtime increases only modestly as more calibration pairs are used.

More Calibration Data Does Not Yield Consistent Gains. Increasing N_{cal} from 10 to 100 slightly improves P@1 from 0.669 to 0.673, but reduces P@5 from 0.871 to 0.859. Further increasing to $N_{cal} = 1000$ recovers and slightly exceeds the default P@5, reaching 0.873, while P@1 remains nearly unchanged at 0.672. This suggests that larger calibration sets do not produce a uniform improvement across metrics.

The Default Setting Remains a Strong Practical Choice. Although $N_{cal} = 1000$ achieves the best P@5, the gain over the default is marginal, and it comes with a small increase in runtime. Meanwhile, the default $N_{cal} = 10$ already delivers competitive P@1 and P@5 with the lowest cost, supporting it as an efficient and reliable default configuration.

Algorithm 1 IToT

Require: Calibration set $\mathcal{D}_{\text{cal}} = \{(x_i, y_i^+)\}_{i=1}^N$, model $\mathcal{M}$, controlled layers $\mathcal{L}_{\text{high}}$, window size W, pool size B, retain size K, momentum β, steering strength $\lambda^{(l)}$, mixing factor α
Ensure: Generated text $\hat{y}$

Phase I: Offline Prototype Calibration
for each $(x_i, y_i^+) \in \mathcal{D}_{\text{cal}}$ **do**
  $y_i^- \leftarrow \mathcal{M}.\text{generate}(x_i)$
  $\{\bar{\mathbf{h}}_i^{(l),+}, \bar{\mathbf{h}}_i^{(l),-}\}_{l \in \mathcal{L}_{\text{high}}} \leftarrow \mathcal{M}.\text{forward}(x_i \oplus y_i^+, x_i \oplus y_i^-)$
end for
for each layer $l \in \mathcal{L}_{\text{high}}$ **do**
  $\mathbf{p}^{(l)} \leftarrow \text{Norm}\left(\sum_{i=1}^N (\bar{\mathbf{h}}_i^{(l),+} - \bar{\mathbf{h}}_i^{(l),-})\right)$
end for
Phase II: Inference-Time Steering
for each layer $l \in \mathcal{L}_{\text{high}}$ **do**
  $\mathcal{B}^{(l)} \leftarrow \emptyset$
  for $b = 1, \ldots, B$ **do**
    $\epsilon_b \sim \mathcal{N}(\mathbf{0}, \sigma^2\mathbf{I})$
    $\mathbf{m}_b^{(l)} \leftarrow \text{Norm}\left(\mathbf{p}^{(l)} + \epsilon_b\right)$
    $v_b^{(l)} \leftarrow 0$
    $\mathcal{B}^{(l)} \leftarrow \mathcal{B}^{(l)} \cup \{(\mathbf{m}_b^{(l)}, v_b^{(l)})\}$
  end for
end for
$\mathcal{Y} \leftarrow [\,]$
while EOS not emitted **do**
  for $w = 1, \ldots, W$ **do**
    $\{\mathbf{h}_t^{(l)}\}_l \leftarrow \mathcal{M}.\text{forward_step}(x, \mathcal{Y})$
    for each layer $l \in \mathcal{L}_{\text{high}}$ **do**
      $b^{*(l)} \leftarrow \arg\max_b v_b^{(l)}$
      $\tilde{\mathbf{h}}_t^{(l)} \leftarrow \mathbf{h}_t^{(l)} + \lambda^{(l)} \|\mathbf{h}_t^{(l)}\|_2 \mathbf{m}_{b^{*(l)}}^{(l)}$
      $\mathbf{h}_t'^{(l)} \leftarrow (1-\alpha)\mathbf{h}_t^{(l)} + \alpha\tilde{\mathbf{h}}_t^{(l)}$
    end for
    $t_{\text{next}} \leftarrow \mathcal{M}.\text{decode_next}(\{\mathbf{h}_t'^{(l)}\})$
    $\mathcal{Y} \leftarrow \mathcal{Y} \oplus [t_{\text{next}}]$
    if $t_{\text{next}} = \text{EOS}$ **then**
      break
    end if
  end for
  for each layer $l \in \mathcal{L}_{\text{high}}$ **do**
    for $b = 1, \ldots, |\mathcal{B}^{(l)}|$ **do**
      $\mathbf{d}_b^{(l)} \leftarrow \text{CMA-ES}\left(\mathbf{m}_b^{(l)}; \text{obj} = \cos(\cdot, \mathbf{p}^{(l)})\right)$
      $\mathbf{m}_b^{(l)} \leftarrow \text{Norm}\left(\beta\mathbf{m}_b^{(l)} + (1-\beta)\mathbf{d}_b^{(l)}\right)$
      $v_b^{(l)} \leftarrow v_b^{(l)} + \cos(\mathbf{d}_b^{(l)}, \mathbf{p}^{(l)})$
    end for
    $\mathcal{B}^{(l)} \leftarrow \text{TopK}(\mathcal{B}^{(l)}, K)$
    Replenish $\mathcal{B}^{(l)}$ to size B
  end for
end while
return $\hat{y} \leftarrow \mathcal{Y}$